Project Name

VANET using Sumo and NS3

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

B.Tech Degree in Computer Science and Technology

Submitted By:
Rajeev Ranjan Sinha (Roll No. 510517035),
Md. Naimuddin Momin (Roll No. 510518007),
Jahnvi Nandan Santoshi (Roll No. 510518021),
Mithun Kumar (Roll No. 510518035),
Aashish Kumar (Roll No. 510518084)

Under the supervision of

Prof. Sipra Das Bit

1856 १८५६
उत्तिष्ठत जाग्रत प्राप्य वरान निबोधत
INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR
भारतीय अभियांत्रिकी विज्ञान एवं प्रौद्योगिकी संस्थान, शिबपुर

December 13, 2021

VANET using Sumo and NS3

Rajeev Ranjan Sinha (Roll No. 510517035),
Md. Naimuddin Momin (Roll No. 510518007),
Jahnvi Nandan Santoshi (Roll No. 510518021),
Mithun Kumar (Roll No. 510518035),
Aashish Kumar (Roll No. 510518084)

Under the supervision of

Prof. Sipra Das Bit

Department of Computer Science and Technology
IIEST, Shibpur
India-711103

Acknowledgment

We would like to express our deep gratitude to Professor Sipra Das Bit, Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, our project supervisor, for her patient guidance, enthusiastic encouragement, and useful critiques for this project.

We would like to thank Professor Asit Kumar Das, the Head of Department, Department of Computer Science & Technology, Indian Institute of Engineering Science and Technology, Shibpur, for his support.

We would also like to thank all the faculty members and staff of the Department, as well as our fellow classmates for their help and support.

(Rajeev Ranjan Sinha - 510517035)

(Md. Naimuddin Momin - 510518007)

(Jahnvi Nandan Santoshi - 510518021)

(Mithun Kumar - 510518035)

(Aashish Kumar - 510518084)

Certificate

This is to certify that the project titled "**Project Name**" has been carried out **by Rajeev Ranjan Sinha (Roll No. 510517035), Md. Naimuddin Momin (Roll No. 510518007), Jahnvi Nandan Santoshi (Roll No. 510518021), Mithun Kumar (Roll No. 510518035), Aashish Kumar (Roll No. 510518084)** under my supervision and guidance in their seventh and eighth semesters in partial fulfillment of the requirements for the B.Tech program in the Department of Computer Science and Technology, IIEST Shibpur. This work has not been reported anywhere for any other purpose.

(Sipra Das Bit)

Abstract

In this report we are demonstrating simulation of Vehicular ad-hoc Network (VANET) using Sumo simulator and NS3. We have used Sumo Simulator to create a traffic scenario with vehicles in it and then the generated file is used by NS3 where we have shown the network simulation of the traffic by NS3. Then with NetAnim we have used the data produced by NS3 to show how the network simulation is actually working(in a graph) with time.

Contents

1 Introduction

1.1 Motivation

Road crashes and the damage they entail represent a serious issue and are one of the main causes of death. Some statistics have shown that the majority of road accidents are due to human error and 60% of these accidents could have been avoided if the driver had been warned at least half a second beforehand. In this context we can design a vehicle-to-vehicle communication system, and with the help of this system vehicles can communicate with each other and reports details of accidents to another vehicle.

1.2 Proposed Project

Design of a Vehicular ad-hoc Network using SUMO simulator and NS2 which can help reduce the road accidents.

1.3 Outline of the Project

Section 1 - Introduction
Section 2 - Discusses all tools and technologies used in the project
Section 3 - Implementations
Section 4 - outlines the work completed till now
Section 5 - outlines the work that can be done in future

2 Tools and Technology

In this section, we will be discussing the tools and technologies used in this project.

2.1 Ubuntu(latest version)

2.2 Sumo Simulator

"Simulation of Urban MObility" (SUMO) is an open source, highly portable, microscopic traffic simulation
package designed to handle large road networks and different modes of transport.

SUMO allows modelling of intermodal traffic systems including road vehicles, public transport and pedestrians. Included with SUMO is a wealth of supporting tools which handle tasks such as route finding, visualization, network import and emission calculation. SUMO can be enhanced with custom models and provides various APIs to remotely control the simulation.

2.3 Open street Map (osmWebWizard.py)

The OSM Web Wizard offers one of the easiest solutions to start with SUMO. Based on a selection of an openstreetmap map excerpt, you will be able to configure a randomized traffic demand and run and visualize the scenario in the sumo-gui.

2.4 NS3

Ns-3 (NETWORK SIMULATOR TOOLS NS3) is a research and educational simulator, by and for the research community. It will rely on the ongoing contributions of the community to develop new models, debug or maintain existing ones, and share results.

This ns3 module implements a **bidirectional coupling to** the road traffic simulator SUMO. It dynamically synchronizes the positions of SUMO vehicles with corresponding ns3 nodes. Additionally, the state of SUMO vehicles can be controlled via ns3, e.g. for changing the speed.

- TCP
- Wi-Fi
- Plotting
- Topology Generators

- Trace analysis
- Mobility scenario generators

❖ What are the mobility models are available in ns3:

- ➢ Supports random mobility models.
- ➢ works inside a rectangular bounded area).
- ➢ Node selects a random direction and speed.
- ➢ Node walks in that direction until the edge.
- ➢ Node pauses for random time.
- ➢ Methods of Mobility Model interfaces:
- ➢ Vector GetPosition().
- ➢ Void SetPosition(Vector pos).

❖ What are the mobility models are available in ns3:

- ➢ Node is at a fixed location; does not move on its own.

❖ Random WayPoint Mobility Model:

- ➢ (works inside a rectangular bounded area).
- ➢ Node pauses for a certain random time.
- ➢ Node selects a random waypoint and speed.
- ➢ Node starts walking towards the waypoint.
- ➢ When the waypoint is reached, goto first state.

3 Implementation

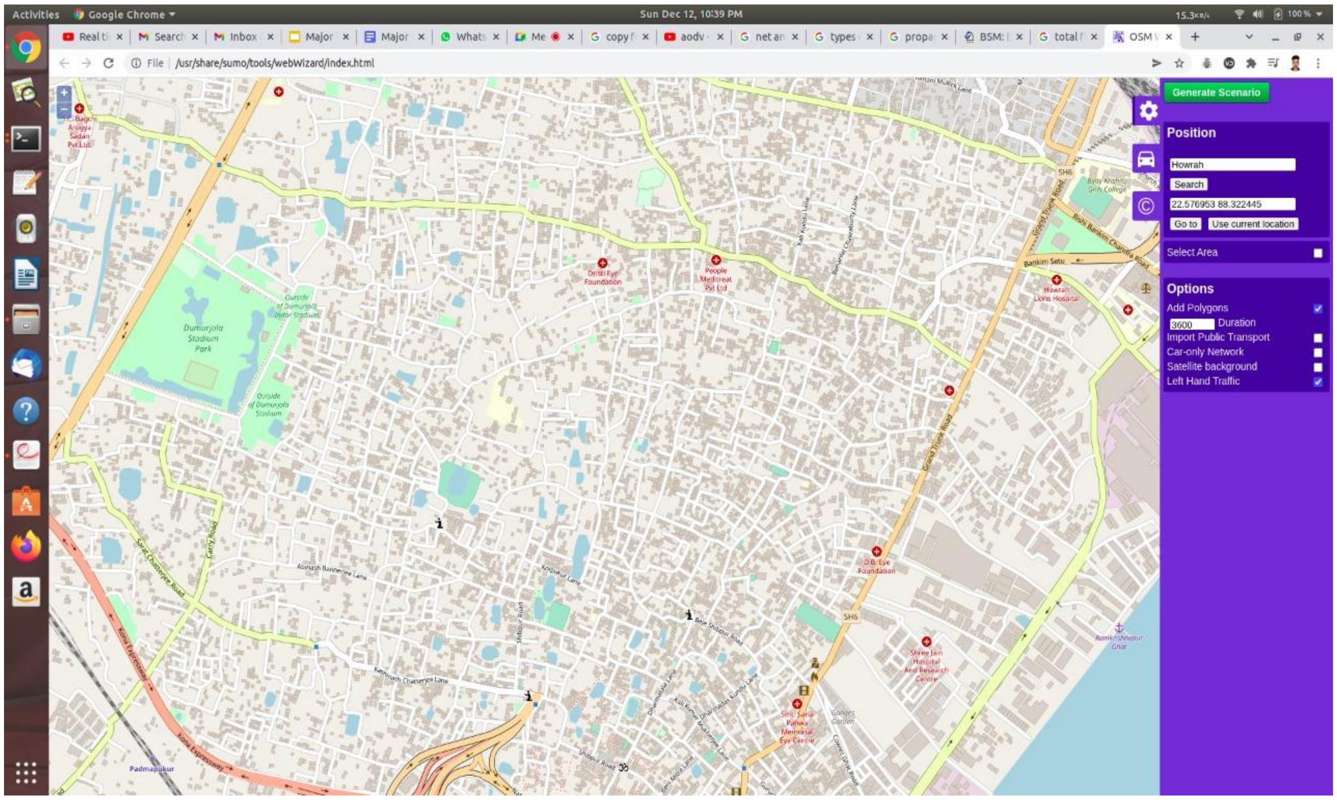
//Step1: Open OSMWebWizard

The OSM Web Wizard is essentially a collection of python scripts located under the directory *tools* in our sumo installation root. We start the OSM Web wizard by invoking the following command in the *tools* directory:

```
$ cd sumo/tools
```

```
$ python osmWebWizard.py
```

Windows users may also invoke the command by clicking *All Programs -> SUMO -> OSM Web Wizard*. Once the script is running, a web browser should open showing a map excerpt of central Howrah.



Position

Howrah [Search]

22.576953 88.322445 [Go to] [Use current location]

Select Area ☐

Options

Add Polygons ☑

3600 Duration

Import Public Transport ☐

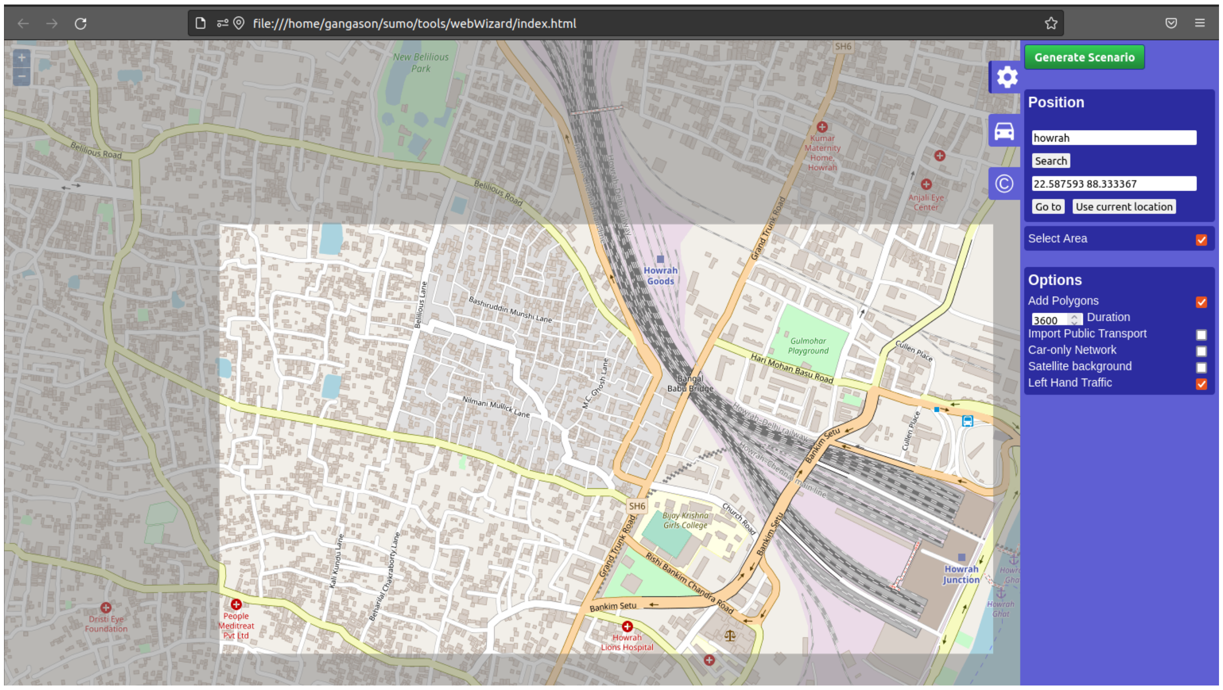
Car-only Network ☐

Satellite background ☐

Left Hand Traffic ☑

We may zoom and pan to the area of our interest. Caution: if the map excerpt covers a very large area, the simulation might become slow or even unresponsive. So we choose a similar zoom level as in the initial view.

In the next step, We select the actual area for which we want to generate the simulation scenario. The area selection will be activated by clicking the checkbox *Select Area* at the blue area selection panel on the right side of the map.



Position
howrah
22.587593 88.333367
Select Area [✓]

Options
Add Polygons [✓]
3600 Duration
Import Public Transport [ ]
Car-only Network [ ]
Satellite background [ ]
Left Hand Traffic [✓]

We can change the size and location of this area by click and hold with the mouse pointer at the boundary between the grayed and non-grayed area. Once we are satisfied with the area selection, We can proceed to the next step.

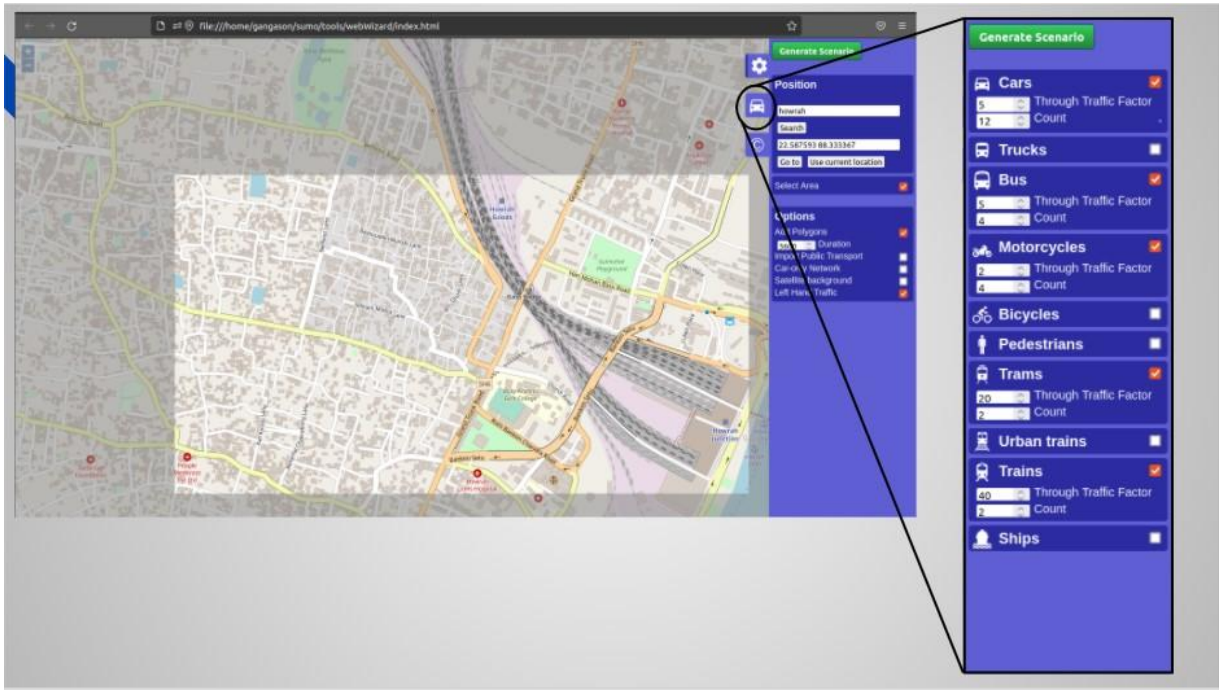
Network Generation

The infrastructure import from OSM into the SUMO simulation are affected by different Wizard options.

- by default a road traffic simulation is generated but all types of roads and rails will be imported as well (cycle paths, footpaths, railways etc).
- if the checkbox "left-hand Traffic" is enabled, the network will be built with left hand traffic rules. For most geographic regions where this is required, this feature will be enabled automatically but if it does not work, this option can be used as a remedy.
- if the checkbox "Car-only Network" is enabled, then only roads that permit passenger car traffic will be included. This can be used to reduce the network size and also helps to reduce intersection complexity.
- if the checkbox "Import Public Transport" is enabled, then busStops and trainStops will be exported. Also busses, trams and trains will be generated that follow the public transport routes defined in OSM (but they will follow synthetic schedules).
- if the Demand-checkbox "Bicycles" is active, then extra bicycle lanes will be added to roads where OSM contains this information.
- if the Demand-checkbox "Pedestrians" is active, then sidewalks and pedestrian crossings will be generated.

Demand Generation

The demand is defined by the demand generation panel. We activate this panel by clicking on the car pictogram.



Cars ✓
5 Through Traffic Factor
12 Count

Trucks ☐

Bus ✓
5 Through Traffic Factor
4 Count

Motorcycles ✓
2 Through Traffic Factor
4 Count

Bicycles ☐

Pedestrians ☐

Trams ✓
20 Through Traffic Factor
2 Count

Urban trains ☐

Trains ✓
40 Through Traffic Factor
2 Count

Ships ☐

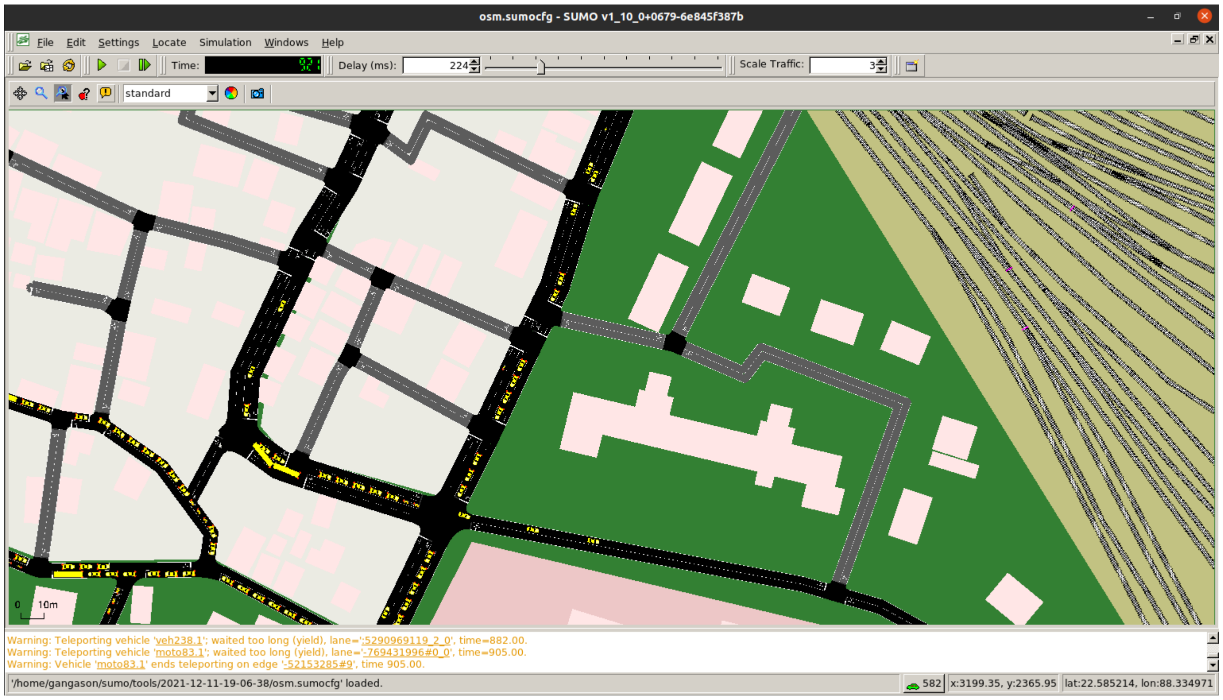
SUMO supports various modes of transport. At the demand generation panel, we can activate/deactivate the individual modes of transport by clicking the corresponding checkboxes. For each mode of transport, the OSM Web Wizard generates random demand based on a certain probability distribution, which is influenced by two parameters:

- Every time a new vehicle is generated, the OSM Web Wizard randomly chooses a departure and arrival edge for the vehicle. The *Through Traffic Factor* defines how many times it is more likely for an edge at the boundary of the simulation area being chosen compared to an edge entirely located inside the simulation area. A big value for the *Through Traffic Factor* implies that many vehicles depart and arrive at the boundary of the simulation area, which corresponds to a scenario with a lot of through traffic.
- The *Count* parameter defines how many vehicles are generated per hour and lane-kilometer.

The next step is generating and running the scenario.

Demand Generation

The complete scenario will be generated automatically once *Generate Scenario* in the control panel has been clicked. The scenario generation takes a couple of seconds or minutes (depending, among others, on the size of the scenario). Once the scenario generation process has finished, the sumo-gui starts and the simulation can be started by pressing the *Play* button.


osm.sumocfg - SUMO v1_10_0+0679-6e845f387b
File Edit Settings Locate Simulation Windows Help
Time:
Delay (ms): 224
Scale Traffic: 3
standard
0 10m

```
Warning: Teleporting vehicle 'veh238.1'; waited too long (yield), lane=':5290969119_2_0', time=882.00.
Warning: Teleporting vehicle 'moto83.1'; waited too long (yield), lane='-769431996#0_0', time=905.00.
Warning: Vehicle 'moto83.1' ends teleporting on edge '-52153285#9', time 905.00.
'/home/gangason/sumo/tools/2021-12-11-19-06-38/osm.sumocfg' loaded.   582   x:3199.35, y:2365.95   lat:22.585214, lon:88.334971
```

//Step2: **Process and find osm.sumocfg and generate vanettrace.xml**

```
$ cd sumo/tools/"folder_name"/

 find osm.sumocfg

$sumo -c osm.sumocfg -fcd-output vanetmobility.xml
```

//Step3: find traceExporter.py and generate vanetmobility.tcl

```
        $ cd sumo/tools

        // find traceExporter.py

        $ python traceExporter.py -i “file_name”/trace.xml
-ns2mobility-output=”file_name”/vanetmobility.tcl

//find total nodes in vanetmobility.tcl
```

//Step4: Add netanimation file

```
#include “ns3/netsnim-module.h”

AnimationInterface anim(“Vanetanim.xml”);      //Add before Simulator::Run();
```

//Step5: Add performance analysis code to vanet program

```
uint32_t SentPackets = 0;

uint32_t ReceivedPackets = 0;

uint32_t LostPackets = 0;

int j=0;

float AvgThroughput = 0;

Time Jitter;

Time Delay;

Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());

  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter = stats.begin (); iter != stats.end (); ++iter)

    {

          Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);

NS_LOG_UNCOND("----Flow ID:" <<iter->first);

NS_LOG_UNCOND("Src Addr" <<t.sourceAddress << "Dst Addr "<< t.destinationAddress);

NS_LOG_UNCOND("Sent Packets=" <<iter->second.txPackets);

NS_LOG_UNCOND("Received Packets =" <<iter->second.rxPackets);

NS_LOG_UNCOND("Lost Packets =" <<iter->second.txPackets-iter->second.rxPackets);

NS_LOG_UNCOND("Packet delivery ratio =" <<iter->second.rxPackets*100/iter->second.txPackets << "%");

NS_LOG_UNCOND("Packet loss ratio =" << (iter->second.txPackets-iter->second.rxPackets)*100/iter->second.txPackets << "%");

NS_LOG_UNCOND("Delay =" <<iter->second.delaySum);
NS_LOG_UNCOND("Jitter =" <<iter->second.jitterSum);

NS_LOG_UNCOND("Throughput =" <<iter->second.rxBytes *
8.0/(iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds())/1024<<"Kbps");

SentPackets = SentPackets +(iter->second.txPackets);

ReceivedPackets = ReceivedPackets + (iter->second.rxPackets);

LostPackets = LostPackets + (iter->second.txPackets-iter->second.rxPackets);

AvgThroughput = AvgThroughput + (iter->second.rxBytes *
8.0/(iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds())/1024);

Delay = Delay + (iter->second.delaySum);

Jitter = Jitter + (iter->second.jitterSum);

j = j + 1;

}

AvgThroughput = AvgThroughput/j;

NS_LOG_UNCOND("--------Total Results of the simulation----------"<<std::endl);

NS_LOG_UNCOND("Total sent packets  =" << SentPackets);

NS_LOG_UNCOND("Total Received Packets =" << ReceivedPackets);

NS_LOG_UNCOND("Total Lost Packets =" << LostPackets);

NS_LOG_UNCOND("Packet Loss ratio =" << ((LostPackets*100)/SentPackets)<< "%");

NS_LOG_UNCOND("Packet delivery ratio =" << ((ReceivedPackets*100)/SentPackets)<< "%");

NS_LOG_UNCOND("Average Throughput =" << AvgThroughput<< "Kbps");

NS_LOG_UNCOND("End to End Delay =" << Delay);

NS_LOG_UNCOND("End to End Jitter delay =" << Jitter);

NS_LOG_UNCOND("Total Flod id " << j);

monitor->SerializeToXmlFile("manet-routing.flowmon", true, true);
```

//Step6: configure scenario 2

//in this scenario we are using the actual data produced by Sumo

```
        else if(m_scenario ==2)

        {

         // Realistic vehicular trace Erode

        // “low density, 2450 total vehicles”

       m_traceFile = “/”filepath”/”foldername”/”filename with tcl extension”;

       m_logfile = “vanet.log”;

       m_mobility = 1;

       m_nNodes = 2450; //The actual number of vehicles

       m_TotalSimTime = 120.01;

       m_nodeSpeed = 0;

      m_nodePause = 0;

      m_CSVfileName = “vanet.csv”;

     m_CSVfileName = “vanet2.csv”;

}
```

//Step7: Run the program

```
$ ./waf –run "scratch/vanet-routing-compare  --scenario=2"
```

//It will give the result of the network performance on the simulation as mentioned in Step 5;

//Step8: Open Netanim

```
$ cd ns-allinone-3.30.1/netanim

$ ./NetAnim
```

Load Vanetanim.xml file and see the network animation

5. Conclusion

With the help of simulation technologies like Sumo and NS3, we created a virtual traffic where we demonstrated how with the help of technology we can actually manage real road traffic problems via network connection where we can get the real time data of each vehicle and monitor the condition of the traffic. This can save a lot of real life problems related to road traffic like jams,accidents, traffic efficiency, and infotainment etc.

6. Future Plan

- Optimising our model to handle a large number of traffic data.
- Broadcasting feedback to other vehicles to diversion and all.
- Broadcasting of Multimedia data along with nodes.
- Using other routing protocols like -
  - ★ OLSR
  - ★ DSDV
  - ★ DSR

  to compare the performance of the network on the simulation.

7. References

1. Sumo Documentation : https://sumo.dlr.de/docs/
2. ScienceDirect Journals : https://www.sciencedirect.com/
3. Youtube Links (provided by Prof. SDB) :
    a. https://youtu.be/IVGgDph51Vg
    b. https://youtu.be/9Tt5GNMuOpc
    c. https://youtu.be/BjU3mdujoXg
4. IEEE Xplore - https://ieeexplore.ieee.org/